Einführung in die Python Programmierung

Fraunhofer Institut für Hochfrequenzphysik und Radartechnik FHR

Praktikumsunterlagen und Aufgabensammlung

Niclas Esser

1 Einleitung

Die Programmiersprache Python wurde von Guido van Rossmann mit dem Ziel entwickelt, um einen einfachen Einstieg in die Welt des Programmierens zu ermöglichen. Tatsächlich wurde die Python zu Schulungs- und Lernzwecke entworfen, denn sie zeichnet sich besonders aus lesbaren und übersichtlichen Code zu schreiben. Der Programmierer ist durch die vorgebene Syntax - auf welcher wir später noch zu sprechen kommen - gezwungen, einen sauberen Stil einzuhalten. Mittlerweile hat Python den vierten Platz der weitverbreitesten Programmiersprachen besetzt, unter den *Skriptsprachen* ist sie sogar die beliebteste. Einer von vielen Gründen der Python so mächtig macht, denn die große Community entwickelt bereits seit fast drei Jahrzenten verschiedenste Bibliotheken mit unterschiedlisten Anwendungsbereichen, es sind (fast) keine Grenzen gesetzt. Beispielweise können Webanwendungen, Datenbank-Verwaltungssysteme, Netzwerkverbindungen, Sensor/Aktorsysteme und mittlerweile sogar Microcontroller progreammiert werden.

1.1 Funktionweise

Python ist wie erwähnt eine Skriptsprache, die sich dadurch auszeichnen, dass der Programmcode nicht kompiliert werden muss, sondern zur Laufzeit *interpretiert* wird. Deshalb ist auch häufig die Rede von Interpretersprachen. Diese unterscheiden sich deutlich von komilierbaren Programmiersprachen wie z.B. Java oder C/C++, bei welchen der Programmcode in ausführbare Dateien (z.B .exe) übersetzt/kompiliert wird. Kompilierte Dateien funktionieren zwar vollständig autonom, aber sind deutlich unflexibler im Vergleich zu Skript- bzw. Interpretersprachen, wie sich später noch herausstellen wird.
Der Python-Interpreter kann auf einfache Weise um neue Funktionen und Datentypen erweitert werden, die in C oder C++ (oder andere Sprachen, die sich von C ausführen lassen) implementiert sind. Auch als Erweiterungssprache für anpassbare Applikationen ist Python hervorragend geeignet.

1.1.1 Datentypen

Wie jede andere Programmiersprache besitzt auch Python Datentypen, die vom Programmierer allerdings nicht deklariert werden müssen. Wir führen einen kleinen Vergleich an dieser Stelle zwischen der Deklaration eines Strings[1] in Java un der eines in Python.

```
//Java
```

[1] Ein String ist eine Zeichenkette die aus Buchstaben, Symbolen, Nummern etc. bestehen kann, jedes einzelne Zeichen ist ein sogenannter *Character* (char).

```java
String str = "1 String in Java!";
char c = "C";
int zahl = 10;
```

```python
//Python
str = "1 String in Python"
c = "C "
zahl = 10
```

2 Praktikum 1

In diesem Teil beschäftigen wir uns mit den Basics der Python Programmierung. In diesem Teil sind Fragen zu beantworten und Programmieraufgaben zu lösen.

2.1 Fragen zu Grundlagen der Programmierung

Im folgenden werden grundlegende Fragen zur Programmierung gestellt. Diese sollen selbstständig beantwortet werden. Falls Du die Fragen nicht beantworten kannst, suche nach der Lösung im Internet

- Was ist ein Bit?

- Was ist ein Byte?

- Was ist ein Integer und wie viel Byte benötigt dieser Datentyp?

- Was ist ein Float und wie viel Byte benötigt dieser Datentyp?

- Was ist ein Double und wie viel Byte benötigt dieser Datentyp?

- Was ist ein Char und wie viel Byte benötigt dieser Datentyp?

- Was ist ein Boolean und wie viel Byte benötigt dieser Datentyp?

- Was ist eine Variable?

- Was ist eine Ausdruck?

- Was ist eine Anweisung?

- Was ist eine Zuweisung?

- Was ist ein Array?

- Was ist eine Funktion?

- Was sind Übergabeparameter und welche zwei Arten gibt es?

- Was ist die Signatur einer Funktion

- Was ist eine main-Funktion?

- Was bedeutet CamelCase?

- Was ist eine IDE und was bedeutet die Abkürzung?

- Was ist ein Textedtior und was ist der Unterschied zur IDE?

- Was versteht man unter Highlighting

- Was bedeutet IntelliSense?

- Was ist die Syntax einer Programmiersprache?

- Was ist eine Datei extension?

- Was ist die Kommandozeile eines Betriebssystems?

- Was ist der Unterschied zwischen intrepretierten und kompilierten Programmiersprache? Nenne jeweils zwei Vorteil!

- Gib einen weiteren Programmiersprachentyp an! Tipp: Gesucht ist ein Typ, der sich besonders durch eine maschinennahe Programmierung auszeichnet.

- Sortiere die folgenden Programmiersprachen in kompilierbare und intrepretierbare: C, C++, Java, Python, PHP, JavaScript, Pearl, GO, MATLAB und LaTeX

Intrepretierbar	Kompilierbar

2.2 Fragen zu Python

Im folgenden werden Python spezifische Fragen gestellt. Diese sollen selbstständig beantwortet werden. Falls Du die Fragen nicht beantworten kannst, suche nach der Lösung im Internet

- Was ist Python

- Was ist Spyder?

- Welche extension trägt eine Python Datei?

- Wie lassen sich Python Dateien von der Kommandozeile ausführen?

- Wodurch wird in Python (automatisch) die Lesbarkeit erhöht?

- Was ist ein Kommentar und wie schreibt man ein solchen in Python?

- Was ein String und wie deklariert man diesen in Python?

- Welche numerischen Datentypen gibt es in Python?

- Lassen sich numerische Datentypen in Python explizit deklarieren?

- Was ist in Python eine Liste und wie deklariert man diese?

- Was ist in Python ein Tuple und wie deklariert man dieses?

- Mit welcher Funktion aus der Python-Standartbibliothek lässt sich Text ausgeben?

- Welche Arten von Schleifen können in Python implementiert werden?

- Wie ist eine for-Schleife in Python defineirt?

- Wie lässt sich in Python ein Array initialisieren?

- Wie lässt sich in Python eine Funktion definieren?

- Was ersetzt in Python die geschweiften Klammern aus Programmiersprachen wie Java und C?

2.3 Die Spyder Python IDE

Bevor wir uns der Programmierung und der tatsächlichen Implementierung von Code widmen, beschäftigen wir uns erst einmal mit der Programmierumgebung. Dieses Praktikum benutzt dazu die Spyder IDE, die Teil des bekanntesten Python-Softwarestacks *Anaconda* ist. Spyder erlaubt uns Code durch Syntaxhervorhebung und Autovervollständigung effzienter und schneller zu programmieren. Des Weiteren stellt die IDE eine Python Konsole zur Verfügung, mit welcher sich "kleinere"Programmabschnitte leicht ausführen lassen. Darüber hinaus ist der Python-Interpreter mit der IDE verknüpft, sodass ausgewählte Dateien und deren Ausgabekommandos zur Laufzeit angezeigt werden können.
Soweit so gut! Bevor wir nun anfangen solltest du aber noch schnell einen Blick in die Dokumentation der Spyder IDE werfen, eventuell ist folgendes Video auch hilfreich.
Links:
- Editor
- Dateiexplorer
- Datei finder- Python Konsole
- Python for Beginners with Spyder IDE (Video)

2.3.1 Vorbereitung

Nun bereiten wir unsere Umgebung vor für die folgenden Programmieraufgaben. Die folgenden fünf Tage werden in 4 Praktika eingeteilt. Dazu erstellen wir nun in einem Arbeitverzeichnis (working-directory). Selbstverständlich könnte man dieses Verzeichnis über die grafische Windows Oberfläche und dem Datei-Explorer erstellen. Da wir aber etwas neues lernen wollen, benutzen wir dafür die Kommandozeile. Drücke die Tasten Win + R , gib `cmd` ein und Enter . Nun erscheint die Kommandozeile. Um sicher zugehen, dass der Python-Interpreter richtig installiert ist, gebe in der Kommandozeile den folgenden Befehl ein und erhalte die folgende Ausgabe:

```
python -V
Python 3.7.3
```

Wobei -V die Flag (Übergabeoption) ist, mit welcher sich die installierte Version anzeigen lässt. Für mehr Informationen über den Python-Interpreter und mögliche Flags gebe diesen Befehl ein:

```
python - -help
```

Sollte keine Ausgabe nach der Befehlseingabe angezeigt werden, du dir aber sicher bist, Python installiert zu haben, liegt es möglicherweise daran, dass die Windows-Umgebungsvariable `PATH` nicht richtig gesetzt wurde. Die Kommandozeile weiß dann nicht in welchem Verzeichnis sie nach den Python-Interpreter suchen soll. Klicke auf den Link, wenn du nicht weißt, wie du die Umgebungsvariable editieren sollst.
Nun erstellen wir das Arbeitsverzeichnis für das Praktikum. Wechsel dazu in der Kommandozeile in das Dokementenverzeichnis. Um Verzeichnisse von der Kommandozeile aus zu wechseln gibt es den Befehl `cd` (cd = change directory). Die Kommandozeile startet immer im Home-Verzeichnis des jeweiligen Nutzers. Da jeder Nutzer einen Dokumenten-Ordner besitzt und dieser ein unmittelbarer Unterordner des Home-Verzeichnis ist, kannst du direkt in diesen wechseln, gebe dazu folgenden Befehl ein:

```
cd Documents
```
oder
```
cd Dokumente
```

Tipp: Um schneller das Verzeichnis wechseln zu können und Tippfehler zu vermeiden nutze die TAB -Taste, diese übernimmt die Autovervollständigung. Den Inhalt eines Ordners kann man sich unter Windows auch mit dem Befehl `dir` ausgeben lassen.
Wenn Du dich jetzt im Dokumenten-Verzeichnis befindest erstelle einen Ordner mit dem Namen `Python_Praktikum` aus der Kommandozeile.

```
mkdir Python_Praktikum
```

Wechsel in den neu erstellten Ordner und erzeuge vier weitere Unterordner mit den Namen `Prakitkum_1, Prakitkum_2, Prakitkum_3, Praktikum_4`

2.4 Aufgaben

Wir haben bereits unser neues Arbeitsverzeichnis eingerichtet. Wir haben auch sichergestellt das Python installiert ist und auch gefunden werden kann. Jetzt machen wir uns an die erste kleine Programmieraufgabe. Öffne dazu die Spyder IDE und suche den Ordner für das erste Praktikum.

2.4.1 Aufgabe 1

Erstelle eine neue Datei (diesmal mit der Spyder IDE) und benenne diese `HelloPython.py`. Öffne diese Datei und erstelle ein Programm, mit der der Ausgabe `Hello Python!`. Tipp: Nutze dafür die Python-Funktion `print()`

2.4.2 Aufgabe 2

Erstelle eine Datei mit dem Namen `math_func.py`. In dieser Datei werden wir mathematiche Funktionen implementieren, die uns später noch zu gute kommen werden.

Aufgabe 2.1

Erstelle zwei Funktionen die als Eingabewerte zwei Variabeln benötigen. Die eine Funktion trägt die Signatur `max(a,b)` und soll den größeren beider Werte zurück geben. Die zweite Funktion trägt die Signatur `min(a,b)` und soll den kleineren Eingabe Wert zurückgeben.

Aufgabe 2.2

Erstelle eine weitere Datei mit dem Namen `test_func.py`, um die bereits programmierten Funktionen zu testen. In dieser Datei werden keine Funktionen implementiert, sondern soll als tatsächliches Skript benutzt werden. Das heißt diese Datei wird letzendlich ausgeführt und ruft die Funktionen der `math_func.py` Datei auf. Prüfe die erstellten Funktionen `max(), min()`, indem numerische Variablen an die Funktion übergeben werden. Bednke, dass der Rückgabewert der Funktionen in weiteren Variablen gespeichert werde sollten, um ein Überschreiben zu verhidnern. Lass dir das Ergebnis ausgeben.
Tipp: Um eine Datei in einer anderen Datei aufzurufen gebe vor dem Programmcode den Ausdruck `import math_func` ein. Anschließend lassen sich alle Funktionen in der importierten Datei aufrufen (Für mehr Informationen klicke hier).

Aufgabe 2.3

Erstelle eine mathematische Funktion die den Betrag von einer komplexen Zahl berechent. Der Betrag einer komplexen Zahl berechent sich über folgende Formel:

$$|z| = \sqrt{a^2 + b^2} \tag{2.1}$$

Das a stellt den Realteil und b den Imaginärteil der komplexen Zahl dar . Die Funktion soll die folgende Signatur tragen `absolut(z)` und den absolut Wert/Betrag zurückgeben. Wenn du nicht genau weißt was eine komplexe Zahl ist klicke hier.
Tipp: Komplexe Zahlen lassen sich in Python so deklarieren `zahl = 3 + 4j`, wobei 3 der Realteil ist und 4 der Imaginärteil (gekennzeichnet durch das "j").

```
num = 3 + 4j     # Zuweisung
r = num.real     # Realteil
i = num.imag     # Imaginaerteil
```

3 Praktikum 2

Erstelle eine Datei mit dem Namen `praktikum2.py` im dafür vorgesehen Ordner `Praktikum_2`. In dieser Datei definieren wir unsere Funktionen und rufen sie ebenfalls dort auf. Das Python Programm bittet den Nutzer einen Wert zwischen 1 - 10 einzugeben. Hierfür kannst du die `input()`-Funktion nehmen. Vergiss aber nicht die Eingabe vom Nutzer zu überprüfen, es dürfen nur Werte zwischen 1 und 10 eingegeben werden.
Je nach Nutzereingabe soll das Programm eine zugehörige Funktion aufrufen, dafür sollst du ein Dictonary anlegen in folgender Form.

```
option{
        0: divBy7
        1: factorial
        2: multipliedDictonary
        3: tupleAndList
        4: InOutStr
        5: calculateFormular
        6: generateTwoDimArray
        7: sortString
        8: upperLetters
        9: removeDuplicates
        10: digitsDividedByFive
}
```

3.1 Aufgabe: Modulo Operator

Schreibe eine Funktion mit zwei Eingabewerten `min` und `max`. Die Funktion soll alle Zahlen finden, die durch 7 teilbar und im Bereich (min/max) liegen. Die Zahlen dürfen nicht durch 2 teilbar sein. Gebe den Zahlen mithilfe eines Strings aus, sodass folgende Beispiel Ausgabe ensteht

```
Choose an exercise (1-15): 1
Set min: 10
Set max: 1000
14, 21, 28, 42 ..., 987, 994
```